Programowanie aspektowe

{ na przykładzie Spring Framework

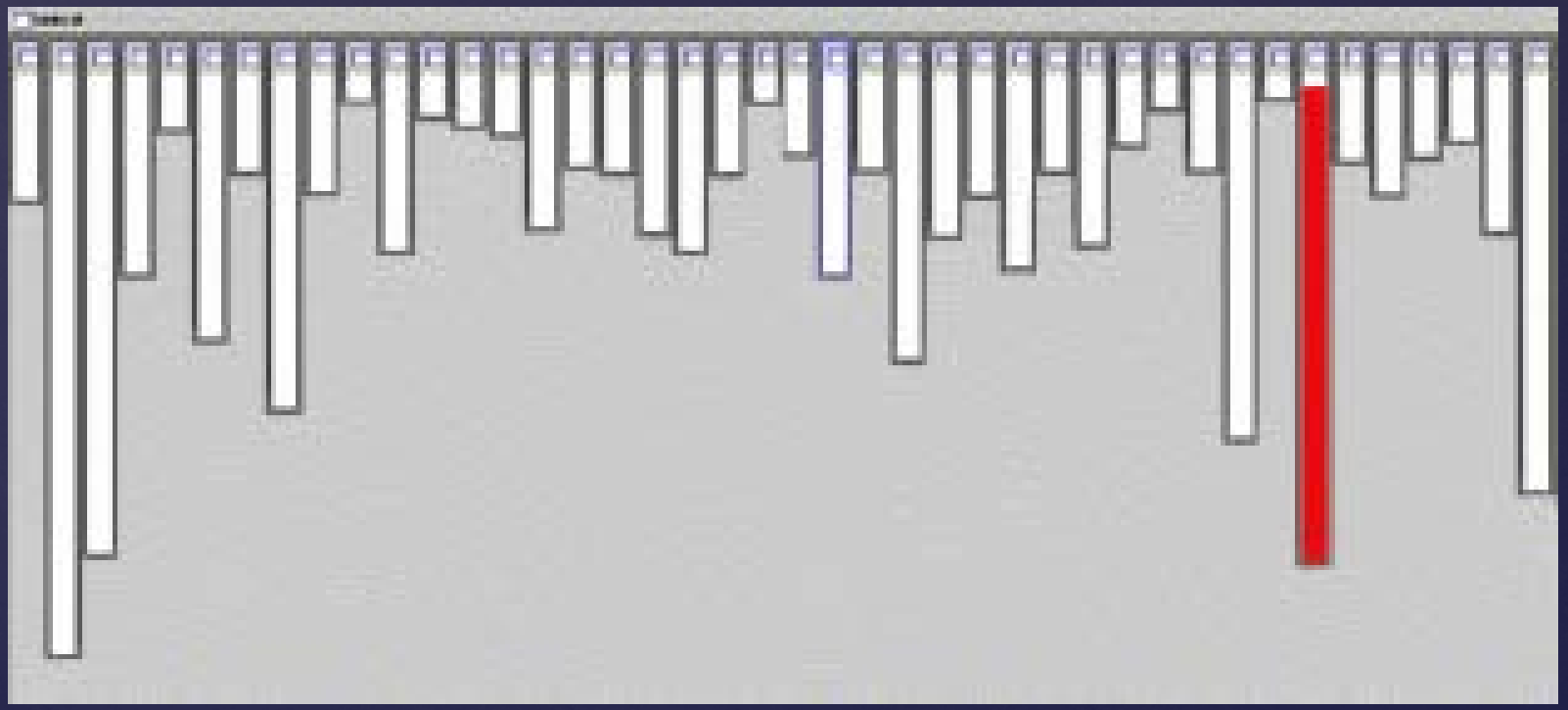
Wszyscy chcemy pisać super kod!

```
public class HelloWorld {
	public static void say(String message) {
		System.out.println(message);
	}
	public static void sayToPerson(String message, String name) {
		System.out.println(name + ", " + message);
	}
}
```

[Źródło: Laddad, I want my AOP!, part 2]

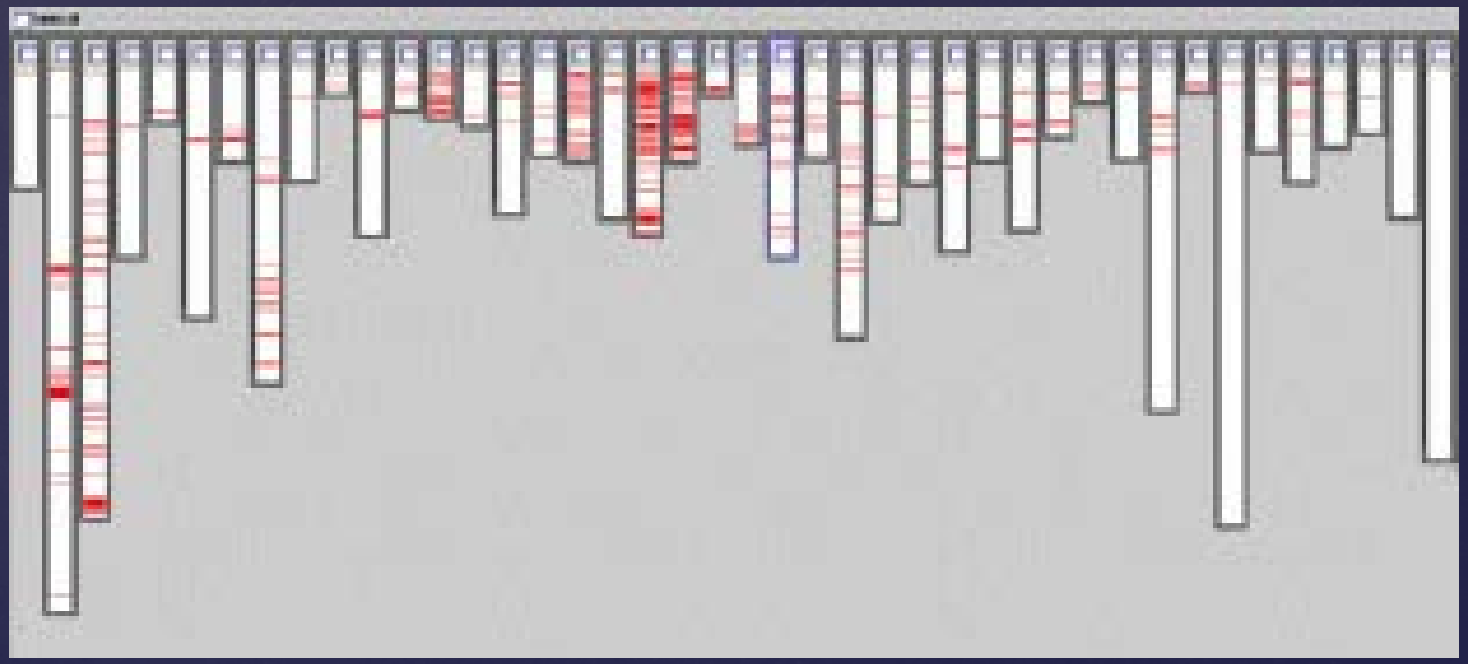
Projekt Apache Tomcat

Realizacja przetwarzania plików XML



Projekt Apache Tomcat – kolejny przykład

Obsługa logowania



Logowanie przykład:

```
public void completeTheTransferMoney(Client client, int num)
{
        super.getLogger().info(„Klient „ + client.getFullName()
            + „chce zalogowac się do systemu”);
        UserLogin log = getUserCredentials(client);
        try {
                log.login();
                // *** wykonaj przelew
                log.logout();
                super.getLogger().info(„Klient „ + client.getFullName()
                    + „ pomyślnie zakończył transakcje przelewu.”);
        } catch(Exception e) {
                super.getLogger().info(„Przerwana transakcja”);
        }
}
```

Czy metoda nie powinna wyglądać tak?

```
public void completeTheTransferMoney(Client client, int num)
{
        // ** wykonaj operację przelewu
}
```

Czego chce od nas klient?

Klient wymaga oprócz realizacji głównego zagadnienia systemu, również inne poboczne np:

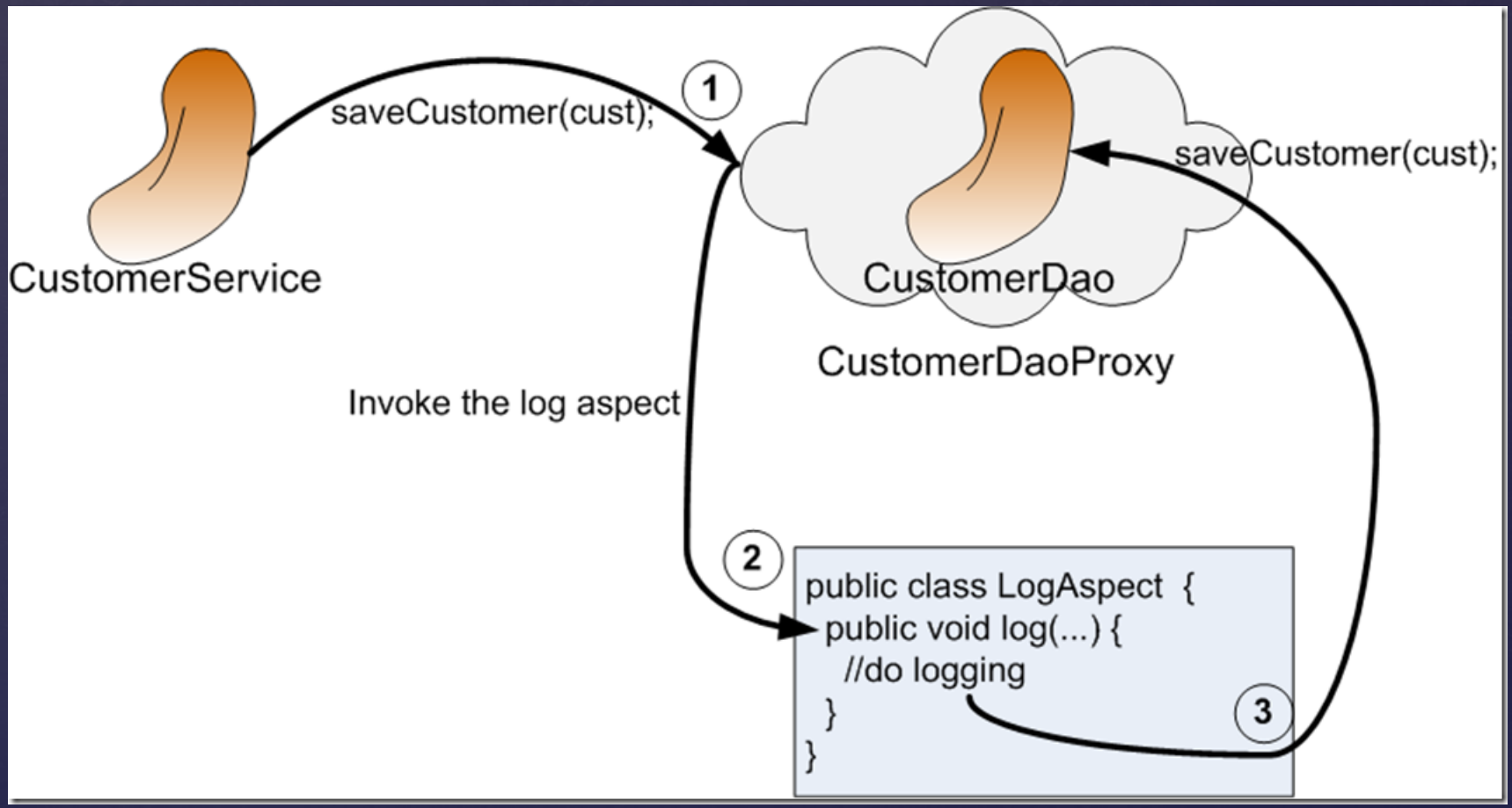
a) logowanie transakcji do dziennika systemowego
b) spójność transakcyjną
c) autoryzację użytkowników
d) rozproszenie obliczeń

AOP nam w tym pomoże!

Anatomia programu aspektowego:

a) Aspekt (aspect)
b) Rada (advice)
c) Punkty łączeń (join points)
d) Linie podziału (pointcuts)

Spring AOP Proxy



Jak to wszystko zrobić?